**3724–3732** *Nucleic Acids Research, 1997, Vol. 25, No. 18*

# Finding the most significant common sequence and structure motifs in a set of RNA sequences

**J. Gorodkin, L. J. Heyer[1] and G. D. Stormo[2,*]**

Center for Biological Sequence Analysis, The Technical University of Denmark, Building 206, DK-2800 Lyngby, Denmark, [1]Department of Applied Mathematics and [2]Department of Molecular, Cellular and Developmental Biology, University of Colorado, Boulder, CO 80309, USA

Received March 27, 1997; Revised and Accepted August 4, 1997

## ABSTRACT

**We present a computational scheme to locally align a collection of RNA sequences using sequence and structure constraints. In addition, the method searches for the resulting alignments with the most significant common motifs, among all possible collections. The first part utilizes a simplified version of the Sankoff algorithm for simultaneous folding and alignment of RNA sequences, but maintains tractability by constructing multi-sequence alignments from pairwise comparisons. The algorithm finds the multiple alignments using a greedy approach and has similarities to both CLUSTAL and CONSENSUS, but the core algorithm assures that the pairwise alignments are optimized for both sequence and structure conservation. The choice of scoring system and the method of progressively constructing the final solution are important considerations that are discussed. Example solutions, and comparisons with other approaches, are provided. The solutions include finding consensus structures identical to published ones.**

## INTRODUCTION

Locating sequence as well as structure motifs in a set of RNA sequences is of general interest. For example, all of the methods that do structure prediction based on phylogenetic data require that the alignment of the sequences be known in advance. That alignment process is usually done by hand and is one of the biggest problems in using that approach. The method presented here promises to automate the alignment and structure determination process, and can be used on normal phylogenetic data, on SELEX (1) type data where the RNAs have been selected *in vitro*, and when one has a collection of genes that are coordinately regulated at the translational level. In contrast to many other RNA folding and aligning methods, we present a method which performs local structural alignment of RNA sequences. The work here is an extended version of (2).

Much work has been put into sequence alignment, e.g. (3,4), including methods to align multiple sequences, e.g. (5–9), but RNA sequences are often conserved more in their structure than in their sequence, so alignments of them based solely on sequence conservation are usually incorrect. There has also been much work on RNA secondary structure prediction, for example through free-energy minimization of structures (10,11). But these methods work on single sequences and are not generally reliable enough to accurately predict the structures of entire sets of RNA sequences. There is a method to build a multiple alignment based on structure predictions of individual sequences (12), but this ignores the sequence component of the alignment. Simulated annealing has also been applied to the problem of aligning multiple RNA structures (13) and to fold individual sequences (14). Also a genetic algorithm has been applied to fold individual sequences (15). It is generally agreed that comparative methods are the most reliable for determining the structure of a set of related RNA sequences (16,17). However, those methods require that the alignment be known in advance. If reasonably good alignments can be obtained, a very effective method is an iterative procedure that uses the alignment to predict the structure, then refines the alignment based on the structure, and repeats the process until no further improvement is seen. However, the actual use of such methods are not very automated and usually require the careful attention of an expert to attain the final resulting alignment. Perhaps the most promising automated approach is the use of stochastic context-free grammars (SCFG) (18,19) which perform global alignment considering both sequence and structure conservation. Eddy *et al.* (18) even show how this method can be used to discover the structure model that is common to the set of sequences. However, these methods do not work well on local alignments. This is the problem we are most interested in, where the conserved motifs represent only a portion of the available sequences. Such is usually the case with *in vitro* selected RNAs, and also with regulatory regions in RNA. In fact, interest in identifying regulatory regions in DNA sequences (20–22) has motivated this work. The problem is more complicated because the motifs being searched for contain structural, as well as sequence, conservation, and also because it is imperative to allow gaps in the alignment. If gaps are not required, then a simple extension to the sequence alignment method can identify the common motifs quite well (23).

With real data there is always the possibility that some of the $N$ sequences are not really related to the rest, but have been included in the set erroneously, or for other reasons, for example

---

[*] To whom correspondence should be addressed. Tel: +1 303 492 1476; Fax: +1 303 492 7744; Email: stormo@colorado.edu

that the $N$ sequences are functionally related but fall into two (or more) structural classes so that there is not a single motif that is common to all of them. In general we want to identify $M \leq N$ sequences that contain the most significant common motif. In our approach that common motif will appear in the alignment of the $M$ sequences, and failure to get good alignments with the rest of the sequences will indicate that they do not belong to the same structural class. However, for $N$ sequences, there are essentially $2^N$ subsets of sequences, and it would not be practical to examine all these subsets to identify the one with the most significant common motif. Thus we include an approach in which poorly aligned subsets are discarded.

In 1985 David Sankoff (24) published an algorithm to optimally fold and align multiple RNA sequences. The main limitation of his algorithm is that its time complexity is $O(L^{3N})$ for $N$ sequences of length $L$. That makes it impractical for any but the smallest problems. However, it is easily computed for two sequences of moderate length. Therefore, we have adapted an approach used in pairwise progressive alignment methods, such as CLUSTAL (5,6), in which each step of the procedure requires alignment of only two entities. Each entity can be either an individual sequence or an alignment of sequences. We also use the strategy of the greedy algorithm in CONSENSUS (21), of maintaining many intermediate solutions so as to minimize the likelihood of missing the optimal solution. While this approach does not guarantee that we will find the optimal solution, as does the Sankoff algorithm, it does have one advantage (besides tractability) over that method, which is that it can locate the subset with the most significant core structure.

We have simplified the basic Sankoff algorithm in two important ways. His algorithm minimizes the total cost of the alignment and, simultaneously, the energy of the structures. Because we are specifically looking for local alignments, we adopt the approach of the Smith–Waterman algorithm for local sequence alignments (3). Therefore we maximize a score based on a combination of sequence similarity and structure. For convenience we choose to score the structure based only on the number of base pairs, and not their stacking energies (10). This change allows us to use the basic Sankoff algorithm to find the highest scoring local alignment of the RNA sequences. (See below for further details of the scoring system.)

The second major simplification reduces the time complexity to $O(L^4)$, rather than the $O(L^6)$ for two sequences of the standard algorithm. We do this by not allowing for branching structures. We justify this approach by claiming that we do not expect our program to identify the complete structural motif in a single pass, but rather to give us a good ‘core structural alignment’ which can be used to obtain the complete motif by existing methods. We certainly expect that the motifs may contain branching structures. They may even contain pseudoknots, and if that is true even the complete $O(L^{3N})$ Sankoff algorithm will not find them, nor will SCFG methods. However, given a good alignment of a ‘core’ region of the common motif, an SCFG method can extend and refine that to capture more of the motif. And maximum-weighted-matching (MWM) methods can be used to identify pseudoknots in the aligned sequences (25,26). The most difficult part of the problem, given existing programs, seems to be identifying a good candidate local alignment from which the total motif can be identified. The $O(L^4)$ algorithm seems quite capable of that, and its increased speed (compared to the $O(L^6)$ version) allows us to do many more comparisons between different subsets of sequences.

## DATA SETS

We select four published data sets for investigation, all from SELEX (1) experiments and for which a consensus structure has been proposed. The first set (H1) of RNA was found to bind to the human immunodeficiency virus type 1 rev protein, and consists of three families of hairpin loops (27). The second set (H2) contains a pseudoknot with specific affinity for HIV-1-RT. This data set also contains erroneous sequences which have been found to be retained by nitrocellulose filters in the absence of protein (28). The third set (THEO) of RNA binds to the bronchodilator theophylline and has a conserved sequence pattern and a consensus structure consisting of a circularly permuted and broken hairpin loop (29). In particular it has a conserved CCU bulge. The fourth set (R17) is RNA ligands for the bacteriophage R17 coat protein (30), which has characteristic tetra-loops and requires an A-bulge in the stem. The length of all the sequences is in the range of 30–50 nt, and the data set sizes range from 13 to 36 sequences. For each data set we have used only the variable part of the sequences. If we had included the constant regions our program would simply have aligned by them. Since, for some of the selected sequences, the common motif overlaps the constant region, these will not be included in the subset of sequences identified by the program. However, they can easily be identified by later searching for the motif in the entire sequence, including the constant region.

## METHOD

The basic principles derived by Sankoff (24) are simplified and extended to perform alignment between two collections (entities) of already aligned sequences, merging them into a new collection of sequences. We extend the 2-D dynamic programming to 4-D dynamic programming which includes folding and show that this is similar to the 2-D case and can be done with a similar score matrix. Finally we add the feature of pruning away low score collections (filtering) from the set of all collections of aligned sequences. The collections with highest scores are the ones used to suggest consensus structures for the sequences within the given collection.

### From 2-D to 4-D dynamic programming for comparing two sequences

For comparison with our new algorithm, we briefly describe the algorithm of Smith and Waterman (3) for finding the maximum scoring local alignment between two sequences, $\vec{a} = (a_1 ...., a_n)$ and $\vec{b} = (b_1, ...., b_m)$. This algorithm essentially comprises the alignment part of our algorithm. We are given a ‘scoring matrix’, $A_{ab}$, which defines the similarity between any two bases, $a$ and $b$. The scoring matrix also includes scores (or penalties) for aligning a base with a gap, $A_{a-}$ and $A_{-b}$. (To simplify the notation, in the following we write $A_{ik}$ to mean $A_{a_i b_k}$, and similarly for gaps, $A_{i-}$ to mean $A_{a_i -}$.) The highest scoring alignment of subsequences is then found using a matrix $\underline{\underline{H}}$, initialized as

$$H_{i0} = H_{0k} = 0, \forall\ 0 \leq i \leq n \text{ and } 0 \leq k \leq m.$$

Subsequent elements of $\underline{\underline{H}}$ are determined by the following recursive relationship:

$$H_{ik} = \max\begin{Bmatrix} H_{i-1,k-1} + A_{ik}\,, \\ H_{i-1,k} + A_{i-}\,, \\ H_{i,k-1} + A_{-k}\,, \\ 0 \end{Bmatrix} \qquad \mathbf{1}$$

[This formulation assumes a constant gap penalty. A more general formulation, allowing arbitrary gap penalties, is similar, but the complexity of the algorithm increases. We actually use an affine gap penalty, with different costs for opening and extending gaps, which does not increase the complexity of the algorithm(31).] $H_{ik}$ is the maximum similarity of two segments ending at $a_i$ and $b_k$. The maximum similarity subsequences end at the maximum element of $\underline{\underline{H}}$, and 'backtracking' to zero then provides the subsequences and their alignment (3). Constraints on the values of $A_{ik}$ are that, on average for random sequences, $A_{ik}$ should be $\leq 0$, and $A_{i-}$ and $A_{-k}$ should be negative and at least equal to the largest difference between a match (i.e. $A_{ii}$) and a mismatch (i.e. $A_{ik}$, $i \neq k$).

Another helpful comparison to our method is the sequence folding algorithm of Nussinov and Jacobson (10). The basic idea of folding is to find the maximum number of basepairs possible for any folding of an arbitrary subsequence ($a_i$, ...., $a_j$). This is done by testing $a_j$'s ability to pair with any base $a_k$ with $i \leq k \leq j-1$. This folding allows for branching structures, and the optimal fold is found from the following recursion:

$$M_{ij} = \max\begin{Bmatrix} M_{i(j-1)} \\ M_{(i+1)(j-1)} + 1 \text{ if } i \text{ and } j \text{ pair.} \\ \max_k \{M_{i(k-1)} + M_{kj}\} \end{Bmatrix} \qquad \mathbf{2}$$

in which each element in the matrix $\underline{\underline{M}}$ is a subsequence of a single sequence. As the algorithm proceeds the maximum number of basepairs for the smaller subsequences has already been computed. The maximal $M_{ij}$ then provides the configuration containing the most basepairs. This algorithm allows for branching configurations, because of the last line (i.e., the $\max_k$ comparisons). By leaving out that line we do not allow for branching structures and we reduce the complexity from $O(N^3)$ to $O(N^2)$. As described above, we assume this faster approach will still find the most significant core structure, and that the remainder of the motif can then be identified by existing methods.

We may now extend the ideas to finding the best subsequence alignment between two sequences, allowing for conserved basepairing within the two subsequences. Formally, we find the best alignment of the subsequences ($a_i$, ...., $a_j$) and ($b_k$, ...., $b_l$), in which the score of the alignment includes terms for sequence similarity between $\vec{a}$ and $\vec{b}$ and basepairing within $\vec{a}$ and $\vec{b}$ that is conserved at aligned positions. Whereas the Sankoff (24) algorithm minimizes a combination of alignment cost and basepairing energy, we maximize the alignment similarity and the number of basepairs that can be simultaneously formed in both sequences. We define a scoring matrix, $S_{ij,kl}$, over all quadruples of bases, including gaps. Details of the scoring matrix are provided below, but letting $D_{ij,kl}$ denote the best score of the subsequences ($a_i$, ...., $a_j$) and ($b_k$, ...., $b_l$), then given any scoring matrix and the constraint of non-branching structures, the maximum scoring subsequence alignments can be obtained from the following recursion:

$$D_{ij,kl} = \max\left\{\begin{array}{ll} D_{(i+1)(j-1),(k+1)(l-1)} + S_{ij,kl}, & (a) \\[6pt] D_{i(j-1),(k+1)(l-1)} + S_{-j,kl}, & \\ D_{(i+1)j,(k+1)(l-1)} + S_{i-,kl}, & (b) \\ D_{(i+1)(j-1),k(l-1)} + S_{ij,-l}, & \\ D_{(i+1)(j-1),(k+1)l} + S_{ij,k-}, & \\[6pt] D_{(i+1)(j-1),kl} + S_{ij,--}, & (c) \\ D_{ij,(k+1)(l-1)} + S_{--,kl}, & \\[6pt] D_{(i+1)j,(k+1)l} + S_{i-,k-}, & (d) \\ D_{i(j-1),k(l-1)} + S_{-j,-l}, & \\[6pt] D_{(i+1)j,k(l-1)} + S_{i-,-l}, & (e) \\ D_{i(j-1),(k+1)l} + S_{-j,k-}, & \\[6pt] D_{(i+1)j,kl} + S_{i-,--}, & \\ D_{i(j-1),kl} + S_{-j,--}, & (f) \\ D_{ij,(k+1)l} + S_{--,k-}, & \\ D_{ij,k(l-1)} + S_{--,-l} & \end{array}\right. \qquad \mathbf{3}$$

The maximal $D_{ij,kl}$ provides the most similar subsequences between the two sequences. One difference between this recursion and that of $\underline{\underline{H}}$, above, is that there is no zero value. This is because the matrix $\underline{\underline{D}}$ indicates both ends of the alignment, $a_i$ with $b_k$ and $a_j$ with $b_l$, and also because we have to allow for negative values occurring within the complete alignment. For example, a hairpin loop might have no similarity between the two subsequences, and therefore have a negative score by itself, yet be contained within the overall best alignment. Since the recursion progresses 'outward' from the hyper-diagonal of the $D$ matrix, those negative values must be kept in order to know the true score of the complete alignment. The similarity to the scoring scheme in the 2-D case is clear. In the 4-D case, each cell of the matrix has 15 neighbors ($2^4 - 1$ 'corners') along which the alignment can proceed. In the 2-D there are only three ($2^2 - 1$ 'corners'). The letters (a)–(f) to the right of the equation separate them into different classes, depending on the number and distribution of gaps that are added to the alignment. One example from each class is shown in Figure 1. Only class (a) gives rise to a structural contribution, and then only if $i$ is complementary to $j$ and $k$ is complementary to $l$. This is because we only score basepairs that are conserved in both sequences.

From the definition of $S_{ij,kl}$ (see next section) it follows that we only need to take the maximum of cases (a), (d) and (f) as the remaining cases can be constructed from these [assuming that the alignment of two –'s (gaps) gets a score of 0]. For example, an alignment of class (b) can be obtained by an alignment of class (f) followed by an alignment of class (d). This reduces the scheme to only seven cases.

The recursion in Figure 1 (equation **3**) is systematically applied to all possible combinations of $i$, $j$, $k$ and $l$ in the two sequences by considering a window of varying length along each sequence, as illustrated in Figure 2. Since the subsequence comparisons in

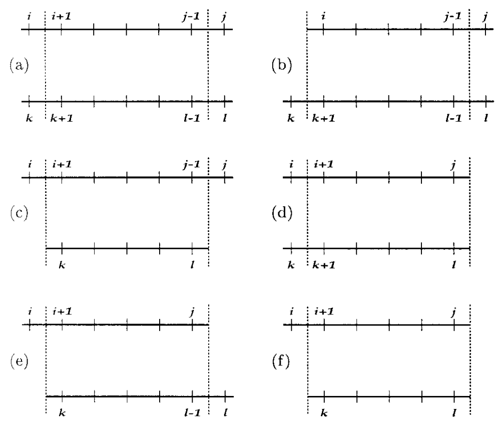

**Figure 1.** Inside the vertical dashed bars is the section that is already structurally aligned, and corresponds to the *D* portion of the right side of equation **3**. The bases listed outside of the dashed bars are the new ones being aligned, with each other or with gaps, and the combined score from all of them is included in the *S* portion of the recursion equations. The cases illustrated as (**a**)–(**f**) correspond to the first example in each class listed in equation **3**.

larger windows use the results of comparisons in smaller windows, each comparison score is stored for later use in the matrix $D_{ij,kl}$. As in other dynamic programming approaches, upon completion of the recursive computation of scores, the maximum element in $\underline{\underline{D}}$ is the starting point for retracing steps. However, now the backward trace continues until the interior of the subsequence is found; that is, until the previous subsequence consists of a single base.

## The scoring matrix for aligning and folding

Consider the score matrix for structurally aligning a sequence $\vec{a}$ against a sequence $\vec{b}$. As before, we reserve the indices *i* and *j* for respective positions in sequence $\vec{a}$, and the indices *k* and *l* for respective positions in sequence $\vec{b}$. For structural alignment between *a* and *b* we construct the total score for each quadruplet (*i*,*j*,*k*,*l*) from two independent contributions, one for sequence

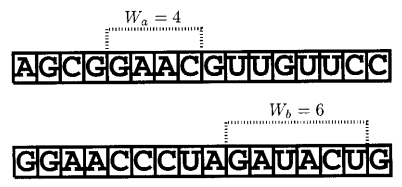

**Figure 2.** The scenario of computing the score between all size windows against all size windows. Here the sliding window of sequence $\overline{a}$ is of size $W_a$ = 4, and for sequence $\overline{b}$, $W_b$ = 6.

alignment and one for alignment of basepairs. We define the score matrix $\underline{\underline{S}}$ to be the sum of matrices $\underline{\underline{A}}$ (alignment) and $\underline{\underline{B}}$ (basepairing)

$$\underline{\underline{S}} = \underline{\underline{A}} + \underline{\underline{B}} \qquad 4$$

which can be written in a simple way using the classic alignment matrix and a matrix defining weighted base complementarity. The score $\underline{\underline{S}}$ is then a list of scores for substituting any pair of bases for any pair of bases (including gaps). The program allows one to define easily the scoring matrix to be used. We have found the following scores to work well, and those have been used for all of the examples in this paper (see equation **5** below):

This matrix gives a score of +4 for basepairing and alignment matches, and a score +3 for any other basepairing without alignment matches (i.e. for compensating changes). Those are the two highest scores obtainable. All other scores are derived from considering only the matching and mismatching of the aligned positions. Note that the computation of score between two given subsequences for which the respective positions are considered fixed (gaps might have been included), may be done by first computing the score for sequence alignment, then by computing the score for structure alignment.

To construct the alignment contribution of this 25 × 25 matrix, we first generalize the classic alignment approach to include alignment of any two bases against any two bases (including gaps). In the standard method of aligning two sequences one defines a similarity of substitutions for some position *i* in the one

$$\underline{\underline{S}} = \begin{array}{c|ccccccccccccccccccccccccc|cc}
 & & & & & & & & & & & & & & & & & & & & & & & & & & \text{k:} & \text{l:} \\
 & 2 & -2 & -2 & -2 & -3 & -2 & -6 & -6 & -6 & -7 & -2 & -6 & -6 & -6 & -7 & -2 & -6 & -6 & -6 & -7 & -3 & -7 & -7 & -7 & -8 & \text{A} & \text{A} \\
 & -2 & 2 & -2 & -2 & -3 & -6 & -2 & -6 & -6 & -7 & -6 & -2 & -6 & -6 & -7 & -6 & -2 & -6 & -6 & -7 & -7 & -3 & -7 & -7 & -8 & \text{A} & \text{C} \\
 & -2 & -2 & 2 & -2 & -3 & -6 & -6 & -2 & -6 & -7 & -6 & -6 & -2 & -6 & -7 & -6 & -6 & -2 & -6 & -7 & -7 & -7 & -3 & -7 & -8 & \text{A} & \text{G} \\
 & -2 & -2 & -2 & 4 & -3 & -6 & -6 & 3 & -2 & -7 & -6 & 3 & -6 & 3 & -7 & 3 & -6 & 3 & -2 & -7 & -7 & -7 & -7 & -3 & -8 & \text{A} & \text{U} \\
 & -3 & -3 & -3 & -3 & 1 & -7 & -7 & -7 & -7 & -3 & -7 & -7 & -7 & -7 & -3 & -7 & -7 & -7 & -7 & -3 & -8 & -8 & -8 & -8 & -4 & \text{A} & - \\
 & -2 & -6 & -6 & -6 & -7 & 2 & -2 & -2 & -2 & -3 & -2 & -6 & -6 & -6 & -7 & -2 & -6 & -6 & -6 & -7 & -3 & -7 & -7 & -7 & -8 & \text{C} & \text{A} \\
 & -6 & -2 & -6 & -6 & -7 & -2 & 2 & -2 & -2 & -3 & -6 & -2 & -6 & -6 & -7 & -6 & -2 & -6 & -6 & -7 & -7 & -3 & -7 & -7 & -8 & \text{C} & \text{C} \\
 & -6 & -6 & -2 & 3 & -7 & -2 & -2 & 4 & -2 & -3 & -6 & 3 & -2 & 3 & -7 & 3 & -6 & 3 & -6 & -7 & -7 & -7 & -3 & -7 & -8 & \text{C} & \text{G} \\
 & -6 & -6 & -6 & -2 & -7 & -2 & -2 & -2 & 2 & -3 & -6 & -6 & -6 & -2 & -7 & -6 & -6 & -6 & -2 & -7 & -7 & -7 & -7 & -3 & -8 & \text{C} & \text{U} \\
 & -7 & -7 & -7 & -7 & -3 & -3 & -3 & -3 & -3 & 1 & -7 & -7 & -7 & -7 & -3 & -7 & -7 & -7 & -7 & -3 & -8 & -8 & -8 & -8 & -4 & \text{C} & - \\
 & -2 & -6 & -6 & -6 & -7 & -2 & -6 & -6 & -6 & -7 & 2 & -2 & -2 & -2 & -3 & -2 & -6 & -6 & -6 & -7 & -3 & -7 & -7 & -7 & -8 & \text{G} & \text{A} \\
 & -6 & -2 & -6 & 3 & -7 & -6 & -2 & 3 & -6 & -7 & -2 & 4 & -2 & 3 & -3 & 3 & -2 & 3 & -6 & -7 & -7 & -3 & -7 & -7 & -8 & \text{G} & \text{C} \\
 & -6 & -6 & -2 & -6 & -7 & -6 & -6 & -2 & -6 & -7 & -2 & -2 & 2 & -2 & -3 & -6 & -6 & -2 & -6 & -7 & -7 & -7 & -3 & -7 & -8 & \text{G} & \text{G} \\
 & -6 & -6 & -6 & 3 & -7 & -6 & -6 & 3 & -2 & -7 & -2 & 3 & -2 & 4 & -3 & 3 & -6 & 3 & -2 & -7 & -7 & -7 & -7 & -3 & -8 & \text{G} & \text{U} \\
 & -7 & -7 & -7 & -7 & -3 & -7 & -7 & -7 & -7 & -3 & -3 & -3 & -3 & -3 & 1 & -7 & -7 & -7 & -7 & -3 & -8 & -8 & -8 & -8 & -4 & \text{G} & - \\
 & -2 & -6 & -6 & 3 & -7 & -2 & -6 & 3 & -6 & -7 & -2 & 3 & -6 & 3 & -7 & 4 & -2 & 3 & -2 & -3 & -3 & -7 & -7 & -7 & -8 & \text{U} & \text{A} \\
 & -6 & -2 & -6 & -6 & -7 & -6 & -2 & -6 & -6 & -7 & -6 & -2 & -6 & -6 & -7 & -2 & 2 & -2 & -2 & -3 & -7 & -3 & -7 & -7 & -8 & \text{U} & \text{C} \\
 & -6 & -6 & -2 & 3 & -7 & -6 & -6 & 3 & -6 & -7 & -6 & 3 & -2 & 3 & -7 & 3 & -2 & 4 & -2 & -3 & -7 & -7 & -3 & -7 & -8 & \text{U} & \text{G} \\
 & -6 & -6 & -6 & -2 & -7 & -6 & -6 & -6 & -2 & -7 & -6 & -6 & -6 & -2 & -7 & -2 & -2 & -2 & 2 & -3 & -7 & -7 & -7 & -3 & -8 & \text{U} & \text{U} \\
 & -7 & -7 & -7 & -7 & -3 & -7 & -7 & -7 & -7 & -3 & -7 & -7 & -7 & -7 & -3 & -3 & -3 & -3 & -3 & 1 & -8 & -8 & -8 & -8 & -4 & \text{U} & - \\
 & -3 & -7 & -7 & -7 & -8 & -3 & -7 & -7 & -7 & -8 & -3 & -7 & -7 & -7 & -8 & -3 & -7 & -7 & -7 & -8 & 1 & -3 & -3 & -3 & -4 & - & \text{A} \\
 & -7 & -3 & -7 & -7 & -8 & -7 & -3 & -7 & -7 & -8 & -7 & -3 & -7 & -7 & -8 & -7 & -3 & -7 & -7 & -8 & -3 & 1 & -3 & -3 & -4 & - & \text{C} \\
 & -7 & -7 & -3 & -7 & -8 & -7 & -7 & -3 & -7 & -8 & -7 & -7 & -3 & -7 & -8 & -7 & -7 & -3 & -7 & -8 & -3 & -3 & 1 & -3 & -4 & - & \text{G} \\
 & -7 & -7 & -7 & -3 & -8 & -7 & -7 & -7 & -3 & -8 & -7 & -7 & -7 & -3 & -8 & -7 & -7 & -7 & -3 & -8 & -3 & -3 & -3 & 1 & -4 & - & \text{U} \\
 & -8 & -8 & -8 & -8 & -4 & -8 & -8 & -8 & -8 & -4 & -8 & -8 & -8 & -8 & -4 & -8 & -8 & -8 & -8 & -4 & -4 & -4 & -4 & -4 & 0 & - & - \\
\text{i:} & \text{A} & \text{A} & \text{A} & \text{A} & \text{A} & \text{C} & \text{C} & \text{C} & \text{C} & \text{C} & \text{G} & \text{G} & \text{G} & \text{G} & \text{G} & \text{U} & \text{U} & \text{U} & \text{U} & \text{U} & - & - & - & - & - & & \\
\text{j:} & \text{A} & \text{C} & \text{G} & \text{U} & - & \text{A} & \text{C} & \text{G} & \text{U} & - & \text{A} & \text{C} & \text{G} & \text{U} & - & \text{A} & \text{C} & \text{G} & \text{U} & - & \text{A} & \text{C} & \text{G} & \text{U} & - & &
\end{array} \qquad 5$$

sequence and some position *k* in another, listed in a score matrix (called $\underline{\underline{A}}$ in the previous section):

$$\underline{\underline{A}}_0 = \begin{array}{c|ccccc|c} & & & & & & k \\ & 1 & -3 & -3 & -3 & -4 & \mathrm{A} \\ & -3 & 1 & -3 & -3 & -4 & \mathrm{C} \\ & -3 & -3 & 1 & -3 & -4 & \mathrm{G} \\ & -3 & -3 & -3 & 1 & -4 & \mathrm{U} \\ & -4 & -4 & -4 & -4 & 0 & - \\ i & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{U} & - & \end{array} \qquad \mathbf{6}$$

where the gap penalty has been included as the dash (initiation and elongation gaps are simply represented by two different matrices). These values are all smaller than those used in Smith and Waterman (3). That is because we are adding a 'bonus' for basepaired positions, and still want to maintain an average negative score for random alignments. We expand this substitution matrix to include all possible pairs for positions (*i*,*j*) in the one sequence and positions (*k*,*l*) in another sequence by adding the cost $(A_0)_{ik}$ for aligning *i* to *k* to the cost $(A_0)_{jl}$ for aligning *j* to *l*, i.e.,

$$A_{ij,kl} = (A_0)_{ik} + (A_0)_{jl} \qquad \mathbf{7}$$

which constitute the elements of $\underline{\underline{A}}$.

When constructing the score matrix for basepairing one might simply put values directly into the 25 × 25 matrix. However, it can also be constructed from two simple components, a matrix of complementary bases and another type of score matrix. First we define the complementary weight matrix by

$$\underline{\underline{C}} = \begin{cases} \underline{\underline{C}}' \text{ if } j\text{–}i \geq d \text{ and } l\text{–}k \geq d \\ \underline{\underline{0}} \text{ otherwise} \end{cases} \qquad \mathbf{8}$$

where *d* is the minimal allowed loop size (typically 3), and where

$$\underline{\underline{C}}' = \begin{array}{c|ccccc|c} & & & & & & j \\ & 0 & 0 & 0 & 1 & 0 & \mathrm{A} \\ & 0 & 0 & 1 & 0 & 0 & \mathrm{C} \\ & 0 & 1 & 0 & 1 & 0 & \mathrm{G} \\ & 1 & 0 & 1 & 0 & 0 & \mathrm{U} \\ & 0 & 0 & 0 & 0 & 0 & - \\ i & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{U} & - & \end{array} \qquad \mathbf{9}$$

accounts for the basepairings. In this example we weight the basepairing G·U the same as any regular Watson–Crick basepair. To construct the score matrix $\underline{\underline{B}}$ for basepairing, we require that $C_{ij}C_{kl} = 0$ implies that $B_{ij,kl} = 0$. Inspired by the case of alignment one can define a basepairing alignment matrix $\underline{\underline{\zeta}}$ providing the score of replacing a basepair in one sequence with a basepair in another. Then we obtain

$$B_{ij,kl} = \sqrt{C_{ij}C_{kl}}\,\zeta_{ik,jl}, \text{ where } \underline{\underline{\zeta}} = \begin{array}{cc|c} 2 & 5 & j=l \\ 5 & 9 & j \neq l \\ i=k & i \neq k & \end{array} \qquad \mathbf{10}$$

where the square root normalizes $C_{ij}C_{kl}$. The matrix $\underline{\underline{\zeta}}$ must be symmetric, and can be interpretated as the cost of aligning *i* to *k* and *j* to *l*, when both *i* and *j*, as well as *k* and *l* basepair. Thus it is sufficient to define $\underline{\underline{\zeta}}$ only when *i* is complementary to *j* and *k* is complementary to *l*. Our choice of values serves to give a much lower score for basepairing when there are also sequence alignment matches because these already get a high score. Scores

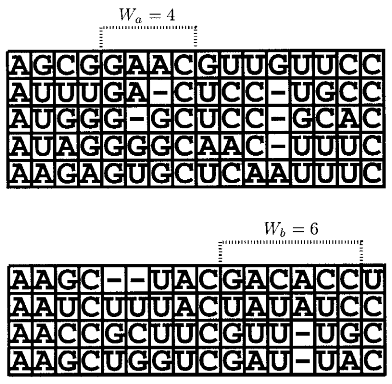

**Figure 3.** The scenario of computing the score between all size windows against all size windows for two collections of aligned sequences. As before the sliding window of sequence $\bar{a}$ is of size $W_a = 4$, and for sequence $\bar{b}$, $W_b = 6$.

for non-matching basepairs are higher because they represent compensating changes where the structure is maintained when the sequences change. The off-diagonal values occur because G·U basepairs allow complementarity to be maintained when only one position changes. The values in $\underline{\underline{\zeta}}$ cause these to be treated as compensating changes in the final $\underline{\underline{S}}$ matrix.

Using the values listed in equations **7** and **10** we obtain the total score matrix for $\underline{\underline{S}} = \underline{\underline{A}} + \underline{\underline{B}}$ listed above, equation **5**.

## Multiple sequence comparison

We now expand the 4-D dynamic programming algorithm to include alignment between two entities or collections of sequences, which do not overlap in the sequences they contain. Consider the case of aligning a collection of *p* aligned sequences to a collection of *q* aligned sequences. The aligned columns within each separate collection remain unchanged, but insertions and deletions can occur between the two sets. The score for the alignment is the sum of all the pairwise scores among all $r = p + q$ sequences. However, we only add in the structural component of the score if every sequence in both sets is complementary at the aligned position pairs. As for the two sequence comparisons, a similar score matrix formalism for multiple sequence comparison has been constructed, but not discussed here. Figure 3 illustrates the computation of the score between two windows from the two collections. The computation is similar to what is done in (6,7).

As extra features we include different penalties for initiation and elongation of gaps. Also we include an optional basepair elongation factor, ε, which scores basepairing such that stacking is favored. When ε = 0 the same number of basepairs throughout an alignment will give the same score no matter where the basepairings are located. When ε > 0 basepairs are scored $B_{ij,kl} + \varepsilon c$, where *c* is the number of nested stacked pairs up to, but not including (*i*,*j*) and (*k*,*l*).

## The greedy algorithm

Here we briefly outline the basic ideas of applying the greedy algorithm to build up multiple alignments. This has similarities to CLUSTALW (6) and CONSENSUS (21). For a set of *N* different sequences there are $2^N - (N + 1) \approx 2^N$ different subsets of two or more sequences. In order to find the best $M \leq N$ aligned sequences, the best scores from many different subsets will have to be compared.

The raw pairwise greedy algorithm is as follows. First all individual sequences are compared to each other, then all the pairwise alignments are compared to all the individual sequences, such that no sequence appears more than once in each comparison. The next greedy step would be to align all the triplet alignments to the individual sequences, and compare all the pairwise alignments to each other, still such that no sequence appears more than once in each final alignment. The algorithm may then be continued until all sequences have been compared in a final alignment. This approach is exponential in time, requiring $O(L^4N^N)$ time. However, many of these comparisons/alignments are redundant and many may also be of such low score that they are unlikely to contribute to the final aligned subset. We can save many comparisons by eliminating alignments that appear unproductive.

Further work is ongoing to optimize the greedy strategy. For the results presented in this paper we have used two means of limiting the number of comparisons. First, one entity is always a single sequence. In the notation introduced above, alignments of $r$ sequences are made by aligning one sequence with $r - 1$ sequences. Second, after the initial alignment of all pairs of sequences, we only store a fixed number of the best scoring alignments, typically 30. For example, we compare each single sequence to each aligned pair of sequences to generate alignments of three sequences. But we only keep the 30 highest scoring alignments for comparison with single sequences to generate the alignments of four sequences. This approach has a time complexity of $O(L^4N^4)$.

Finally, in selecting the 'best' alignment of $M \leq N$ sequences, there is a problem that scores increase with the number of sequences, and so different sized sets are difficult to compare. We need stopping criteria that indicate the largest set of well-aligned sequences occurs at some particular round. As a first approach we look at a few best alignments of each round $r$, and compare score versus alignment length. We find a trade off from which the final alignment is chosen by plotting the scores as a function of $r$, and stopping at the round for which the rate of change decreases, i.e. at the empirical inflection point. Then we look at the five best scored alignments of $r$ and $r - 1$ sequences. Work is ongoing to make the selection of the 'best' alignment more rigorous and more automated.

## RESULTS

For comparison, we attempt to find the alignment for each data set using three publicly available programs. Depending on the data set, and type and amount of conservation, these programs perform with varying degrees of success. None of them is consistently able to identify the structural motif common to the set of sequences as well as our program FOLDALIGN, an implementation of the 4-D alignment and greedy schemes presented here.

### CLUSTALW

We performed multiple alignment of the SELEX data by using the program CLUSTALW (5,6) with default parameters. We used the full dynamic programming alignment between two sequences for the progressive alignment (6,32) performed by CLUSTALW. Since this program performs multiple alignment based on sequence conservation alone, we would not expect it to identify structurally conserved regions. As expected, it does fairly well on data sets with significant amounts of sequence conservation. For example, it properly aligns the conserved hairpin loop, and the conserved bulged A, in the R17 data (33), but fails to align consistently the conserved stem region. It also adds spuriously aligned bases near the 5′ end of the sequences that are not part of the functional motif. The H2 data (28) also contain significant sequence conservation in the first stem of a pseudoknot, and CLUSTALW aligns that region well, although with no suggestion of the conserved structure. The THEO data (29) contains sequence conservation, but it is misaligned in the two subclasses. The H1 data (27) has less sequence conservation than the others and is not well aligned by CLUSTALW.

### COVE: covariance models of RNA

The COVE program by Eddy and Durbin (18) finds tree representations of secondary structures by applying a dynamic programming algorithm to find pairwise mutual information scores of all nucleotide positions. A covariance model is then derived from the resulting optimal tree representation of the structure. The algorithm is also similar to proposed stochastic context-free grammar models developed by the Haussler group (19,34). COVE performs global alignment on a collection of sequences and has been shown to perform well on tRNA for which a global (consensus) structure is defined. As options in the program one can either construct a model from a set of sequences with known structure and apply it to find structures for new sequences, or utilize it to suggest a common structure for a set of sequences. Additional features, like using alignment of the sequences, are also available. It is well known that it is hard to find local features using a global alignment procedure, so we did not expect this package to perform well on the SELEX data sets, and it did not. Using the same data sets as for CLUSTALW we did not find any strong signals for consensus structures, not even if we used the CLUSTALW alignments as input to the program. The multiple structural alignment method presented here can be used to construct a core model which can then be used by COVE to extend and refine that model.

### Tools from the Vienna RNA package

The Vienna RNA package by Hofacker *et al.* (12) includes a program RNAfold to find the minimum free energy structures, and a program RNAdistance to find the distance between two structures in terms of the smallest cost along the editing path when representing the structures as trees. The RNAfold program is based on the work in (11,35,36), and the RNAdistance program on the work in (37–39). We folded each sequence in the data sets and found that many of the structures resembled the published structures, when neglecting the H2 pseudoknot. Using RNAdistance to pairwise compare the structures and appropriate cut-off scores to select reasonable comparisons, we found a number of sequences with similar structures (in the respective data sets) were clustered together. However, we did not find a strongest common structure, or the consensus structure for the respective data sets.

**Table 1.** The strongest aligned class of the H1 data set

| | |
|---|---|
| GGAUUUGAGAUACAC-GGAA-GUGGACUCUCC | 17 |
| GCC-UUGAGAUACACUAUAUAGUGGAC-CGGC | 5 |
| GGC-UGGAGAUACAAACUAU-UUGG-CUCGCC | 4a |
| AUU---GAGAAACAC-GUUU-GUGGACUCGGU | 6b |
| ACC-UUGAGGUACUC-UUAA-CAGG-CUCGGU | 11 |
| GCA-UUGAGAAACAC-GUUU-GUGGACUCUGU | 6a |
| GAA-UUGAGAAACAC--UAA-CUGGCCUCUUU | 14 |
| (((.........((........)).....))) | (publ.) |
| (((...(.(...((........))..).)))) | (FOLDALIGN) |

The parentheses indicate basepairing. The numbers refer to the published sequence labels. Only the aligned part of the sequences are shown

**Table 2.** The strongest aligned class of the H2 data set

| | |
|---|---|
| CCAGAGGCCCAACUGGUAAACGGGC | 1.17 |
| CCG-AAGCUCAACGGGAUAAUGAGC | 2.4a |
| CCG-AAGCCGAACGGGAAAACCGGC | 1.3a |
| CC-CAAGCGC-AGGGGAGAA-GCGC | 1.6 |
| CCG-ACGCCA-ACGGGAGAA-UGGC | 1.8 |
| CCGUUUUCAG-UCGGGAAAAACUGA | 1.1 |
| CCGUUACUCC-UCGGGAUAAAGGAG | 2.11 |
| CCGUAAGAGG-ACGGGAUAAACCUC | 2.7a |
| CCG-UAGGAG-GCGGGAUAU-CUCC | 2.10 |
| CCG--UGCCG-GCGGGAUAU-CGGC | 1.9b |
| CCG-AACUCG-ACGGGAUAA-CGAG | 2.1b |
| CCG--ACUCG--CGGGAUAA-CGAG | 2.12 |
| [[....((((....]].....)))) | (publ.) |
| ......((((...........)))) | (FOLDALIGN) |

The parentheses indicate basepairing (and the square brackets for pseudoknot). The numbers refers to the published sequence labels. Only the aligned part of the sequences are shown.

**Table 3.** FOLDALIGN for the THEO set

| | |
|---|---|
| AUACCAGUGACAACU-CUCGAG-AUCACCCUUGGAAG | TCT8-6,9 |
| AUACCAUCGUGUAAG-CAAGAG-CACGACCUUGGCAG | TCT8-5 |
| AUACCAACUA---CUCUCAC-A-AUAGUCCUUGGAAG | TR8-8 |
| AUACCAACGG----C-AUAU-UUGCUGUCCUUGGAAG | TR8-14 |
| AUACCAACAG----C-AUAU-UUGCUGUCCUUGGAAG | TCT8-1,10 |
| AUACCAGCAU----C-GUCU-U-GAUGCCCUUGGCAG | TCT8-4,8 |
| ...(((((((..............))))...)))... | (publ.) |
| .(.(((((((....(.(....).)))))...))).). | (FOLDALIGN) |

## FOLDALIGN

We determined structural alignments for each of the data sets and compared them to the published consensus structures. For each of the investigated data sets we did find the subset with the strongest common structure, matching what has been published. The first data set, H1 (27), consists of 20 sequences and has been assigned three structural classes, the classes all containing the same structural elements. The largest class consists of 10 sequences, however three of them use the constant SELEX regions (see ref. 1) in the basepairing (which were not included in our data sets), so we would not expect to find more than seven of them with conserved structure. For those seven sequences FOLDALIGN finds the proper alignment and consensus structure matching the published one. This is shown in Table 1. The additional structure identified by FOLDALIGN (Table 1), but not included in the published consensus structure, is included in the consensus structure for the largest class (figure 3 of ref. 27). FOLDALIGN has captured the interacting bases in an internal loop. Furthermore FOLDALIGN succeeds in merging the two strongest classes, missing one sequence in the largest class and getting only one sequence wrong in the alignment (not shown). All of the sequences in the third class utilize part of the constant region in their structures, and so cannot be aligned properly with the other sequences when using only the variable region.

Even better results are obtained for the data set H2 (28). As mentioned this data set contains a pseudoknot, two overlapping stem–loop regions, and therefore violates the knot constraint in the dynamic programming. One stem region is highly conserved in sequence, and the other has almost no sequence conservation. FOLDALIGN aligns the sequence conserved regions based on their sequence alignment, but at the same time aligns the other stem region which is only conserved in structure (see Table 2). This is a good example of the application of the method. Programs like MWM (25,26) could refine the alignment to identify the entire motif, including the pseudoknot.

**Table 4.** FOLDALIGN for the R17 set of the sequences which have at least six basepairs in the stem

| | |
|---|---|
| UGCGCACCAUCAGGGCGU | 12 |
| AAUGCACCAUCAGGGCAU | 6 |
| AUGUUACCAUCAGGAACA | 27 |
| UGCAGAGGAUCACCCUGC | 24 |
| AUGUCACGAUCACGGGCA | 17 |
| AGUAGAGGAACACCCUAC | 32 |
| AUUAGAGGAUCACCCUAG | 25 |
| UAUAGAGCAUCAGCCUAU | 21 |
| AAGAUAGCAUCAGCAUCU | 28 |
| .((((.((....)))))) | (publ.) |
| .((((.((....)))))) | (FOLDALIGN) |

**Table 5.** FOLDALIGN for the R17 set of the sequences which have at least five basepairs in the stem

| | |
|---|---|
| CUCACCAUCAGGGGG | 3 |
| CUCACCAUCAGGGGG | 4 |
| AGGACCAUCAGGCCU | 18 |
| CGCACCAUCAGGGCG | 12 |
| UGCACCAUCAGGGCA | 6 |
| GUUACCAUCAGGAAC | 27 |
| CAGAGGAUCACCCUG | 24 |
| GUCACGAUCACGGGC | 17 |
| UAGAGGAACACCCUA | 32 |
| UAGAGGAUCACCCUA | 25 |
| UAGAGCAUCAGCCUA | 21 |
| GAUAGCAUCAGCAUC | 28 |
| (((.((....))))) | (publ.) |
| (((.((....))))) | (FOLDALIGN) |

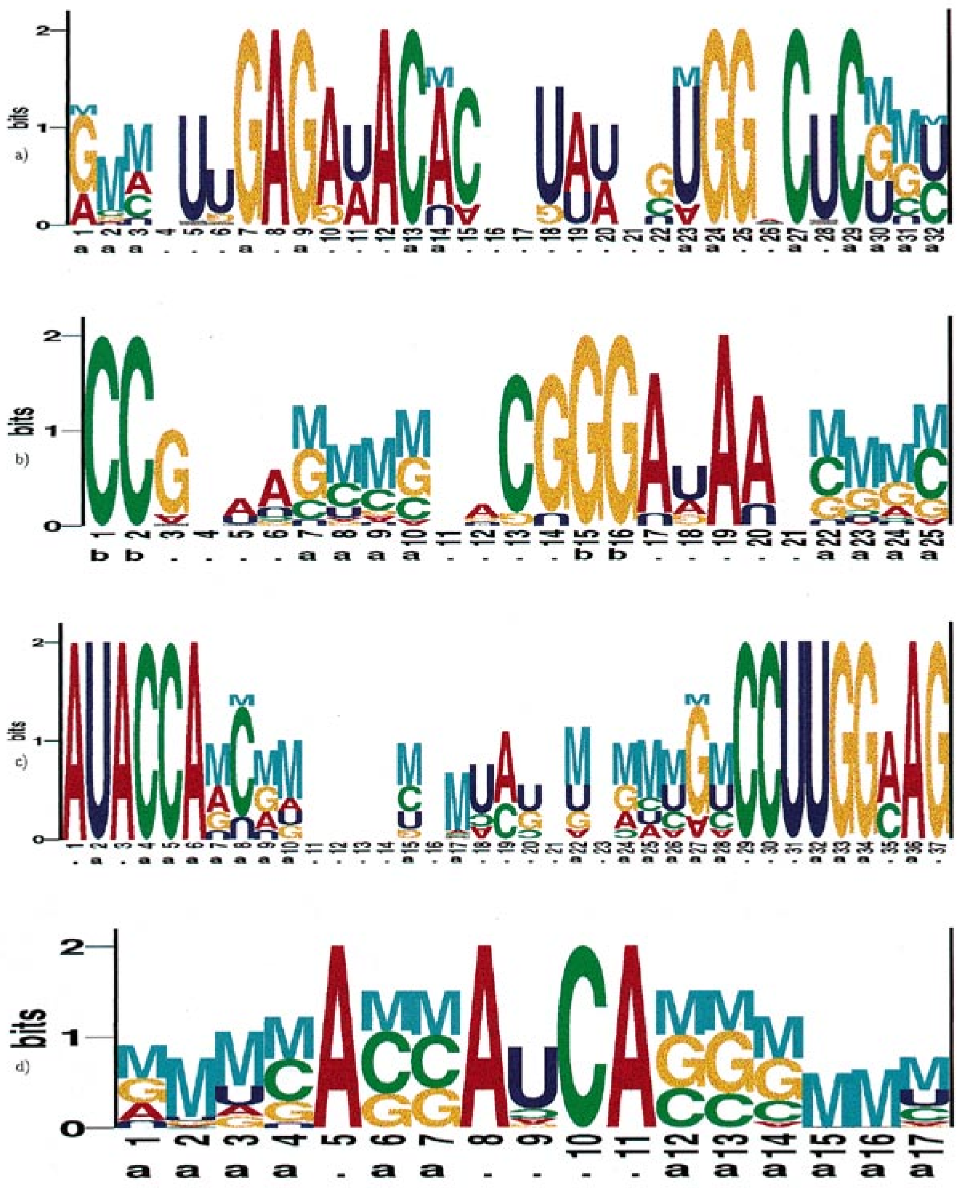

**Figure 4.** The logos for the data. The symbols for sequence alignment are A, C, G, U and –. The letter displaying the mutual information between basepaired positions is M. The sequence-structure logos are for (**a**) the data set H1, (**b**) the data set H2, (**c**) the data set THEO and (**d**) the data set R17. Only positions with positive information content are shown. The letters 'a' and 'b' indicate basepairing for respective regions. For the data set H2 we inserted the known pseudoknot for illustration.

The third data set, THEO (29), consists of two structural classes which are circular permutations of each other. FOLDALIGN identifies the proper motif from the largest class, getting the alignment exactly right for six of the eight sequences (see Table 3). The two remaining sequences contain the shortest stems, only two basepairs in one case, and require the most gaps for proper alignment. The second class could not be aligned with the first due to the circular permutation, but their common structure should be identifiable if they are treated as a separate class (not tested).

The final set, R17 (30), consists of 36 sequences. For many of these, part of the structural motif is contained in the constant region of the sequence, and so is not available to the program for alignment. However, we obtained a perfect alignment for the subset of nine sequences that have at least six basepairs in the stem (Table 4). We also found a perfect alignment for 12 of 16 sequences with at least five basepair stems (Table 5). Alignment of larger subsets are nearly correct, although they do contain a few misaligned sequences. In general, for this data set and the others, the identification of subsets of sequences with conserved core structures should be enough information to extend and refine the complete motif utilizing existing programs mentioned above.

In the results presented above, a few different scoring matrices were used, with essentially the same results for each. Work in progress will further explore various scoring schemes and their impact on resulting alignments. In real time the runs (alignments) were completed in 4–12 h depending on the set sizes. The runs were performed on a Silicon Graphics power challenge IRIX64 machine.

To display the sequence structure content of the alignments we show four examples of structure logos (40) in Figure 4. The structure consists of a sequence part, which is a generalized form of the Schneider and Stephens sequence logo (33). On top of this sequence logo the mutual information between corresponding basepairs has been displayed using the letter ‘M’.

## CONCLUSION

We have presented a method to structurally align a set of RNA sequences, as well as select the subsets containing the most significant alignments. The method, which consists of 4-D dynamic programming and a greedy algorithm for pairwise comparison of sequences, was able to fully find the published alignments of conserved motifs. The complete structure was not always obtained, as in the case of the pseudoknot due to the dynamic programming limitation. But the core alignment that is obtained can be used by existing methods to complete the motif identification. For this type of problem the method clearly outperforms methods based on sequence alignment alone, or structure based methods like stochastic context-free grammars and free energy minimization, the latter including pairwise alignment of structures only. We conclude that our method, to a very large extent, can replace the alignments currently made by hand, or provide significant hints to assist with ‘hands-on’ methods.

Further work is ongoing to improve the identification of the most significant subset alignments. We are also working to fully automate the detection of subclasses of motifs; the current version is really only capable of finding the single most significant common structure, but alignments based on disjoint subsets of the sequences can be utilized for classification into distinct motif classes. The method is also very amenable to parallelization, which should greatly facilitate more extensive comparisons and analyses.

## ACKNOWLEDGEMENTS

Thanks to B.Javornik and NexStar for providing the data electronically. We thank S.Eddy for advice on the use of COVE and an anonymous reviewer for many helpful comments. This work was sponsored in part by NIH grant HG00249 to GDS. JG was supported by the Danish National Research Foundation.

## REFERENCES

1 Tuerk, C. and Gold, L. (1990) *Science* **249**, 505–510.
2 Gorodkin, J., Heyer, L. J., and Stormo, G. D. (1997) In T. Gaasterland, P. Karp, K. Karplus, C. Ouzounis, C. Sander, and A. Valencia (eds), *Proceedings of the Fifth International Conference on Intelligent Systems in Molecular Biology.* AAAI Press, Menlo Park, CA. pp. 120–123.
3 Smith, T. F. and Waterman, M. S. (1981) *J. Mol. Biol.* **147**, 195–197.
4 Waterman, M. S. and Eggert, M. (1987) *J. Mol. Biol.* **197**, 723–728.
5 Higgins, D. G., Bleasby, A.J. and Fuchs, R. (1991) *Comput. Appl. Biosci.* **8**, 189–191.
6 Thompson, J. D., Higgins, D. G. and Gibson, T. J. (1994) *Nucleic Acids Res.* **22**, 4673–4680.
7 Lipman, D., Altschul, S. and Kececioghlu, J. (1994) *Proc. Natl Acad. Sci. USA* **86**, 4412–4415.
8 Baldi, P., Chauvin, Y., Hunkapiller, T. and McClure, M. A. (1994) *Proc. Natl. Acad. Sci. USA* **91**, 1059–1063.
9 Krogh, A., Brown, M., Mian, I. S., Sjölander, K. and Haussler, D. (1994) *J. Mol. Biol.* **235**, 1501–1531.
10 Nussinov, R. and Jacobson, A. B. (1980) *Proc. Natl. Acad. Sci. USA* **77**, 6309–6313.
11 Zuker, M. and Stiegler, P. (1981) *Nucleic Acids Res.* **9**, 133–148.
12 Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M., and Schuster, P. (1994) *Monatshefte für Chemie* **125**, 167–188 (http://www.tbi.univie.ac.at/~ivo/RNA/).
13 Kim, J., Cole, J. R. and Pramanik, S. (1996) *Comput. Appl. Biosci.* **12**, 250–267.
14 Schmitz, M. and Steger, G. (1996) *J. Mol. Biol.* **255**, 254–266.
15 Gultyaev, A. P., vanBatenburg, F. H. D. and Pleij, C. W. A. (1995) *J. Mol. Biol.* **250**, 37–51.
16 Westhof, E. and Michel, F. (1994) In K. Nagai and I. W. Mattaj (eds), *RNA–Protein Interactions.* IRL Press at Oxford University Press, Oxford, UK. pp. 26–51.
17 Westhof, E., Auffinger, P. and Gaspin, C. (1996) In M. J. Bishop and C. J. Rawlings (eds), *DNA–Protein Sequence Analysis.* IRL Press at Oxford University Press, Oxford, UK. pp. 255–278.
18 Eddy, S. and Durbin, R. (1994) *Nucleic Acids Res.* **22**, 2079–2088 (http://genome.wustl.edu/eddy/\#cove).
19 Sakakibara, Y., Brown, M., Hughey, R., Mian, I. S., Sjölander, K., Underwood, R. C. and Haussler, D. (1994) *Nucleic Acids Res.* **22**, 5112–5120.
20 Stormo, G. D. and Hartzell, G. W.,III (1989) *Proc. Natl. Acad. Sci. USA* **86**, 1183–1187.
21 Hertz, G. Z., Hartzell, G. W.,III and Stormo, G. D. (1990) *Comput. Appl. Biosci.* **6**, 81–92.
22 Heumann, J. M., Lapedes, A. S. and Stormo, G. D. (1994) In Altman, R., Brutlag, D., Karp, P., Lathrop, R. and Searls, D. (eds), *Proceedings of the Second International Conference on Intelligent Systems in Molecular Biology.* AAAI Press, Menlo Park, CA. pp. 188–194.
23 Heumann, J. M., Lapedes, A. S., and Stormo, G. D. (1995) In *Proceedings of the 1995 World Congress on Neural Networks II.* INNS Press, Washington, DC. pp. 771–775.
24 Sankoff, D. (1985) *SIAM J. Appl. Math.* **45**, 810–825.
25 Cary, R.B. and Stormo, G. D. (1995) In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. and Wodak, S. (eds), *Proceedings of the Third International Conference on Intelligent Systems in Molecular Biology.* AAAI Press, Menlo Park, CA. pp. 75–80.
26 Tabaska, J. E. and Stormo, G. D. (1997) In T. Gaasterland, P. Karp, K. Karplus, C. Ouzounis, C. Sander, and A. Valencia (eds) *Proceedings of the Fifth International Conference on Intelligent Systems in Molecular Biology.* AAAI Press, Menlo Park, CA. pp. 311–318.
27 Tuerk, C., MacDougal, S., Hertz, G. Z. and Gold, L. (1992) In F. Ferré, K. Mullis, R. Gibbs, and A. Ross (eds), *The Polymerase Chain Reaction.* Birkhauser, Springer-Verlag, NY.
28 Tuerk, C., MacDougal, S., and Gold, L. (1992) *Proc. Natl. Acad. Sci. USA* **89**, 6988–6992.
29 Jenison, R. D., Gill, S. C., Pardi, A. and Polisky, B. (1994) *Science* **263**, 1425–1429.
30 Schneider, D., Tuerk, C., and Gold, L. (1992) *J. Mol. Biol.* **228**, 862–869.
31 Gotoh, O. (1982) *J. Mol. Biol.* **162**, 705–708.
32 Feng, D.-F. and Doolittle, R. F. (1987) *J. Mol. Evol.* **25**, 351–360.
33 Schneider, T. D. and Stephens, R. M. (1990) *Nucleic Acids Res.* **18**, 6097–6100.
34 Sakakibara, Y., Brown, M., Underwood, R. C., Mian, I. S. and Haussler, D. (1994) In L. Hunter (ed.), *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences: Biotechnology Computing, vol. V.* IEEE Computer Society Press, Los Alamitos, CA. pp. 284–293.
35 McCaskill, J. S. (1990) *Biopolymers* **29**, 1105–1119.
36 Turner, D. H., Sugimoto, N. and Freier, S. M. (1988) *Annu. Rev. Biophys. Biophys. Chem.* **17**, 167–192.
37 Shapiro, B. A. (1988) *Comput. Appl. Biosci.* **4**, 381–393.
38 Shapiro, B. A. and Zhang, K. (1990) *Comput. Appl. Biosci.* **6**, 309–318.
39 Fontana, W., Konings, D. A. M., Stadler, P. F. and Schuster, P. (1993) *Biopolymers* **33**, 1389–1404.
40 Gorodkin, J., Heyer, L. J., Brunak, S. and Stormo, G. D. (1997) *Comput. Appl. Biosci.*, in press (http://www.cbs.dtu.dk/gorodkin/appl/slogo.html).